

CSE211 Digital Design

Akdeniz University

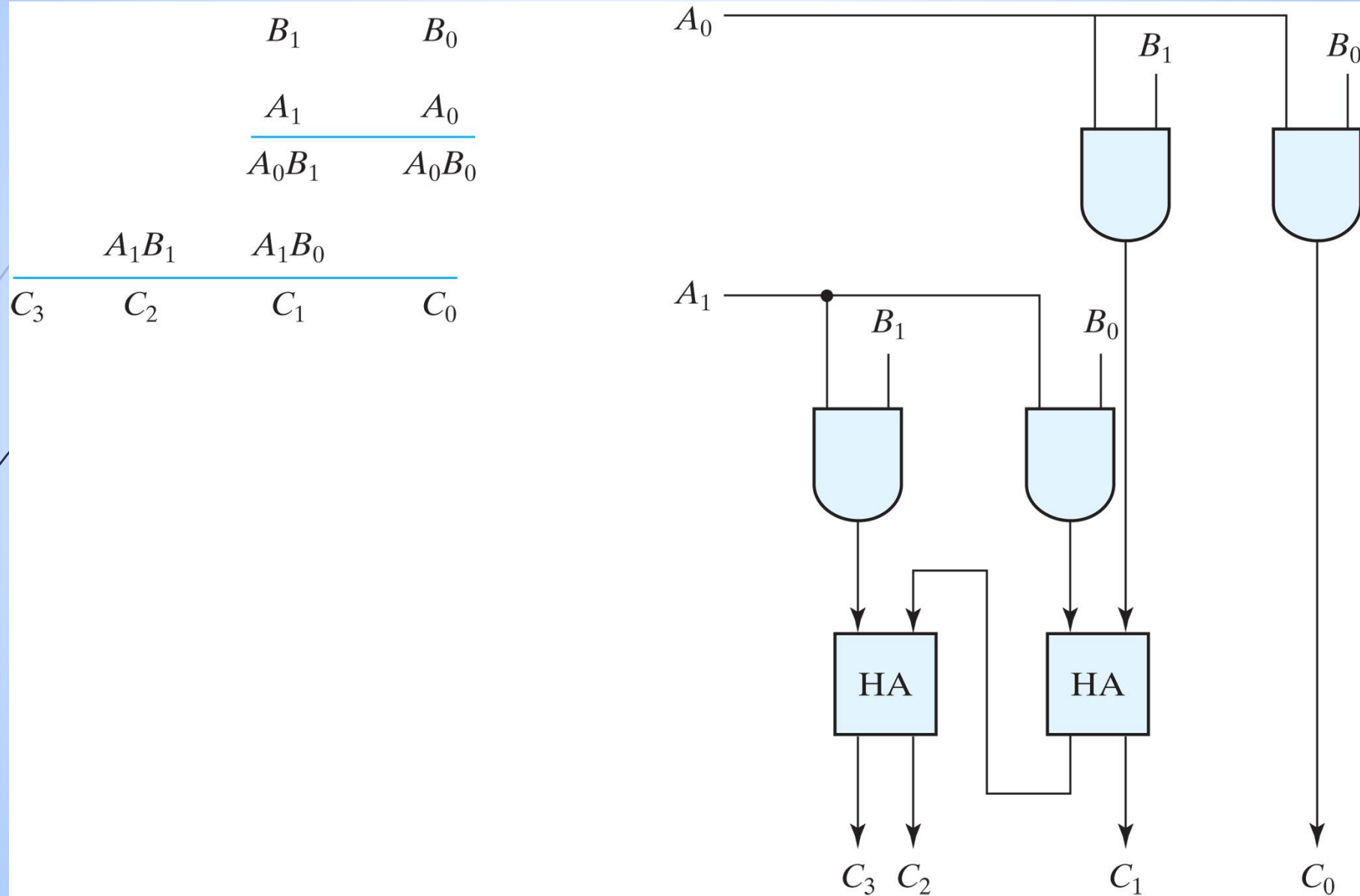
Week12: Combinational Logic Part 2

Assoc.Prof.Dr. Taner Danişman

tdanisman@akdeniz.edu.tr

Binary Multiplier

2



Binary Multiplier

3

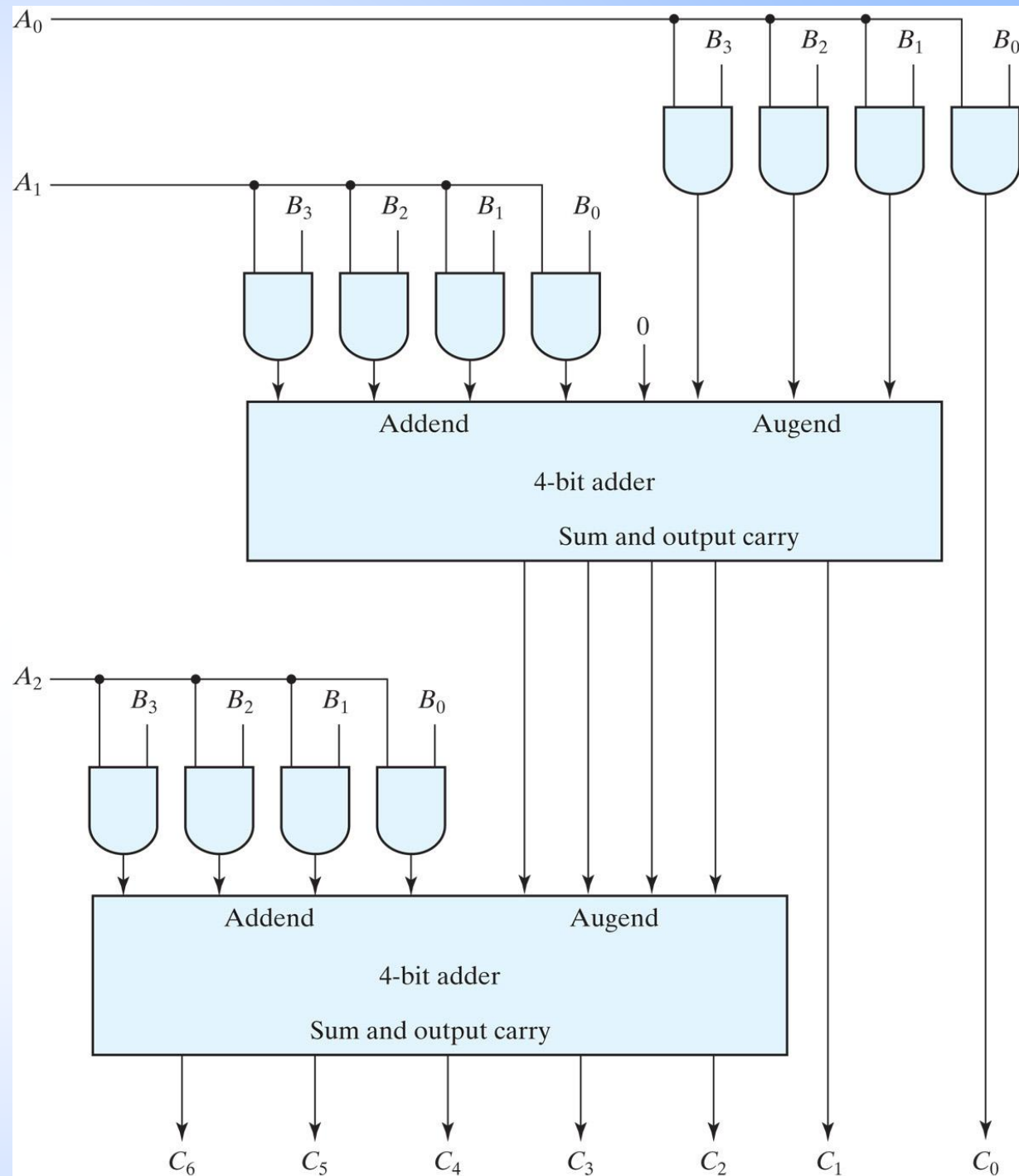
- For **J** multiplier bits and **K** multiplicand bits, we need
 - $(J \times K)$ AND gates and
 - $(J - 1)$ **K -bit adders** to produce a product of $(J + K)$ bits.

MULTPLICAND	→	6
MULTIPLIER	×	3
<hr/>		
PRODUCT	→	18

4

Four-bit by Three bit multiplier

- $B_3B_2B_1B_0$ by $A_2A_1A_0$
- Since $K = 4$ and $J = 3$, we need **12 AND gates** and **two 4-bit adders** to produce a product of **seven bits**



Magnitude Comparator

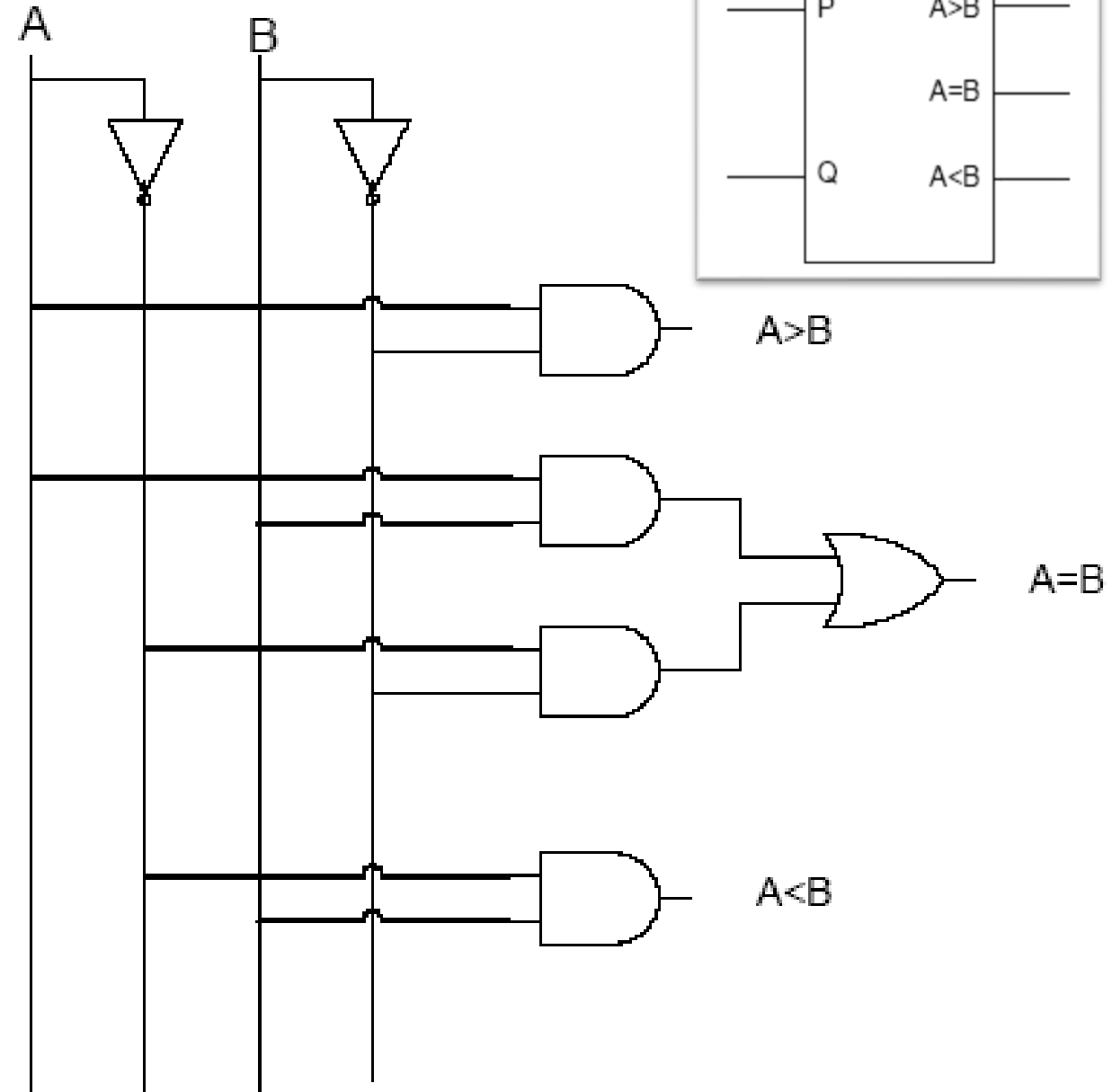
5

- The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number.
- **Magnitude comparator** is a combinational circuit that compares two numbers A and B and determines their relative magnitudes.
 - $A > B$
 - $A = B$
 - $A < B$

Magnitude Comparator

6

A	B	A>B	A=B	A<B
0	0			
0	1			
1	0			
1	1			



Magnitude Comparator

7

- The circuit for comparing two n -bit numbers has **2^{2n} entries** in the truth table and becomes too cumbersome, even with $n = 3$.

- $A=B$ (X-NOR Solution)

$$x_i = A_i B_i + A'_i B'_i \quad \text{for } i = 0, 1, 2, 3$$

$$(A = B) = x_3 x_2 x_1 x_0$$

- To determine whether A is greater or less than B , we inspect the relative magnitudes of pairs of significant digits, starting from the most significant position.

Magnitude Comparator

8

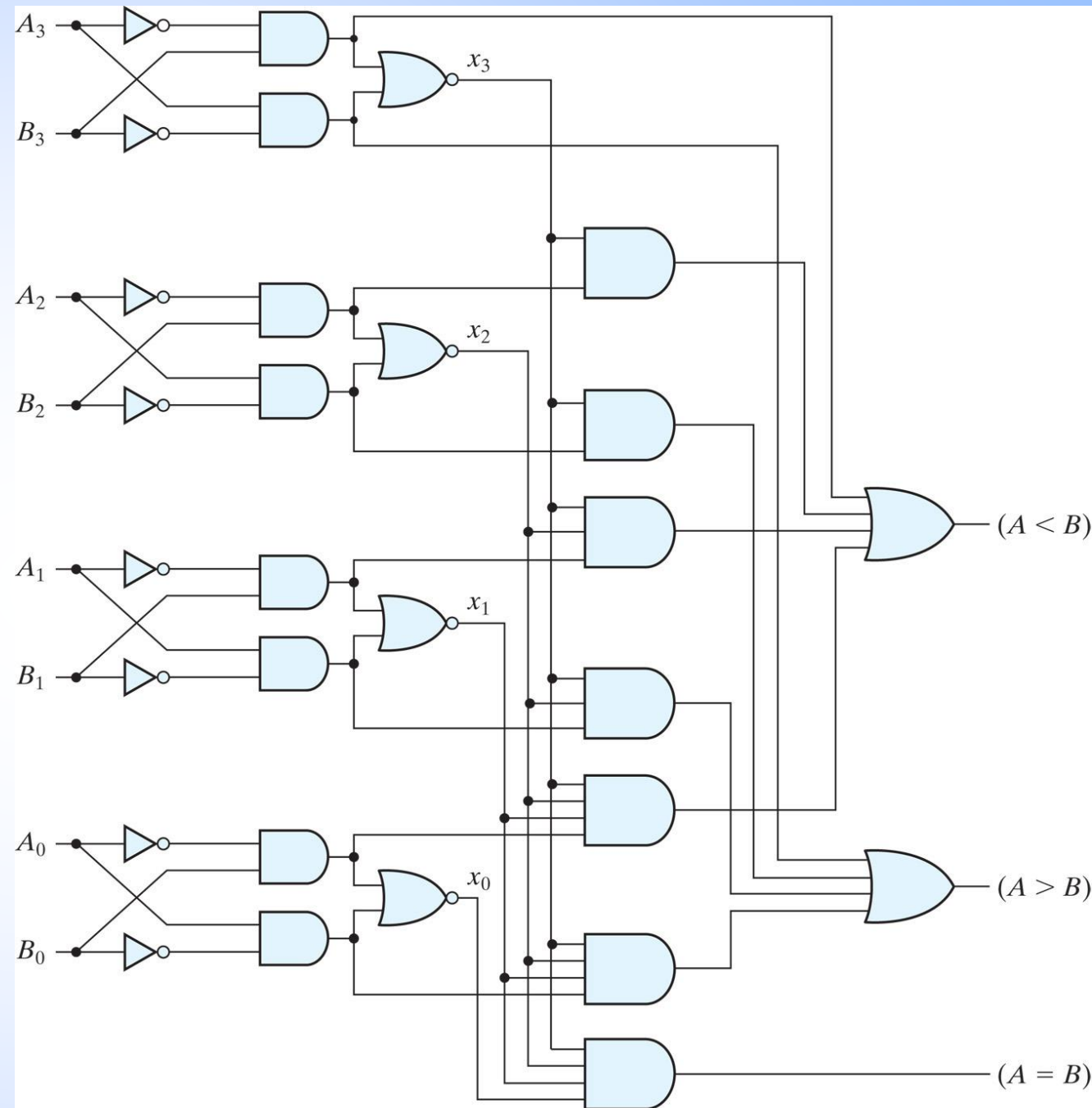
- Greater than and less than

$$(A > B) = A_3B'_3 + x_3A_2B'_2 + x_3x_2A_1B'_1 + x_3x_2x_1A_0B'_0$$

$$(A < B) = A'_3B_3 + x_3A'_2B_2 + x_3x_2A'_1B_1 + x_3x_2x_1A'_0B_0$$

- The symbols $A > B$ and $A < B$ are *binary* output variables that are equal to 1 when $A > B$ and $A < B$, respectively.

4-bit Magnitude Comparator



Decoders

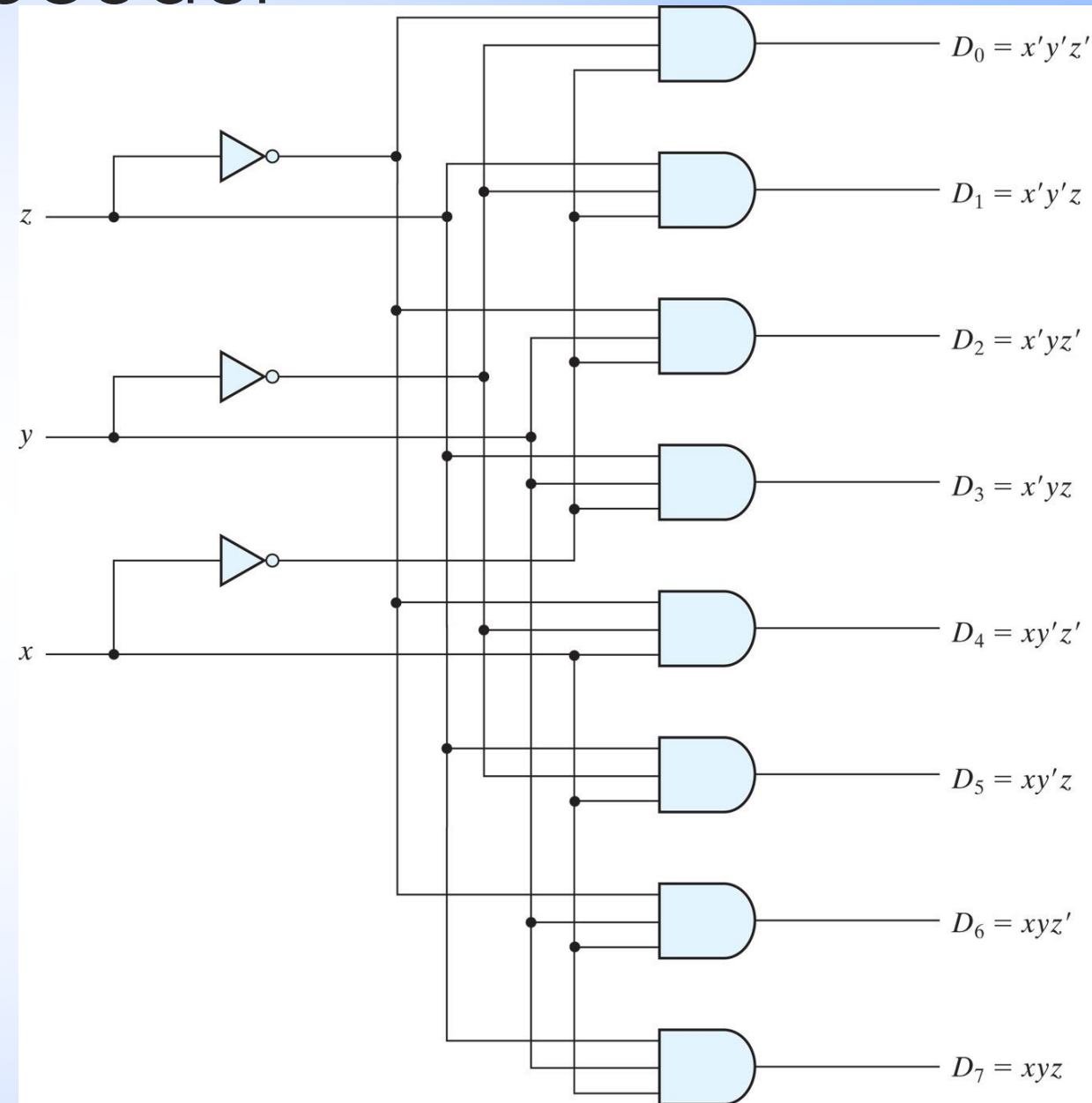
10

- ▶ A binary code of n bits is capable of representing up to 2^n **distinct elements** of coded information.
- ▶ A **decoder** is a **combinational circuit** that converts binary information from n input lines to a maximum of 2^n unique output lines.
- ▶ If the n -bit coded information has **unused** combinations, the decoder may have fewer than 2^n outputs.
- ▶ Their purpose is to generate the 2^n (or fewer) **minterms** of n input variables
- ▶ The name **decoder** is also used in conjunction with other code converters, such as a **BCD-to-seven-segment decoder**

Three-to-eight-line decoder

11

- ▶ A particular application of this decoder is binary-to-octal conversion



Three-to-eight-line decoder

12

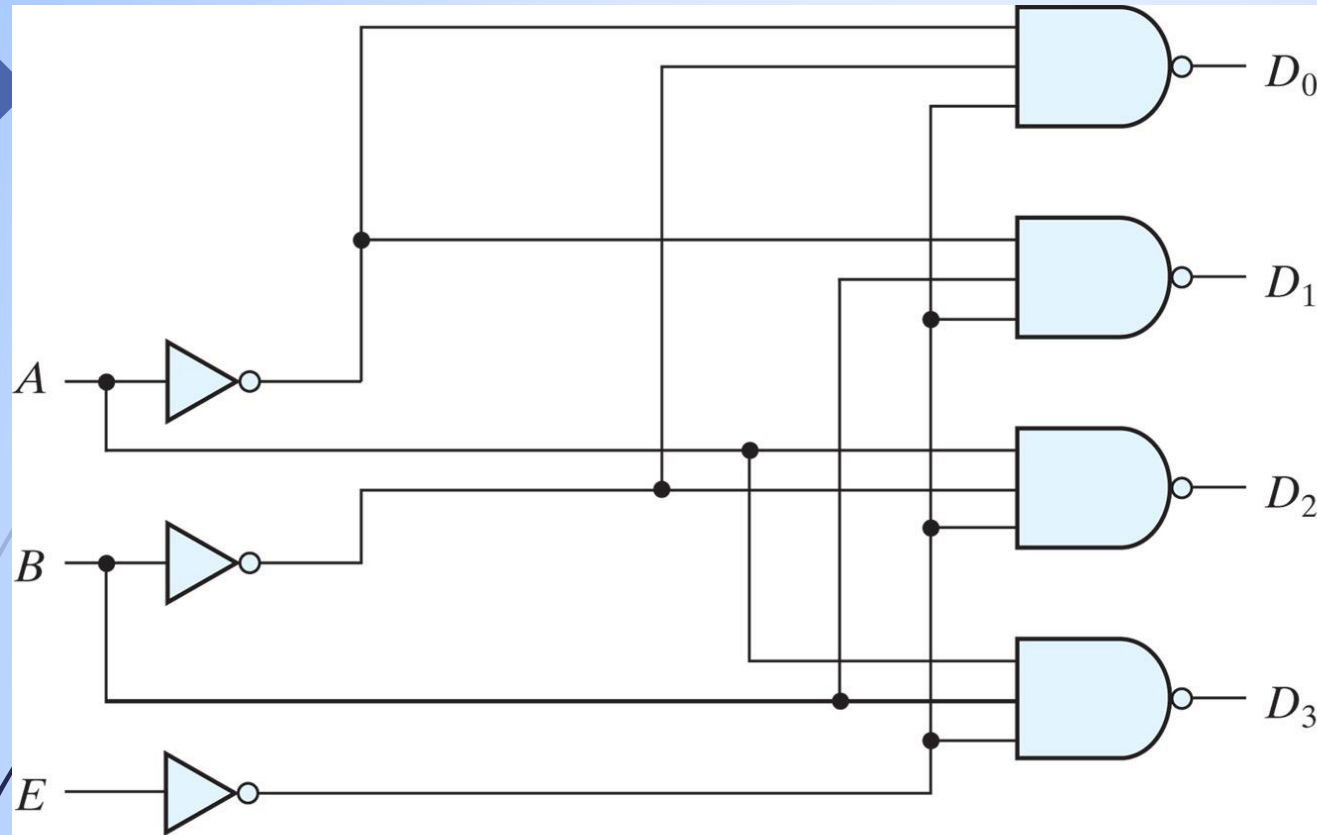
Table 4.6

Truth Table of a Three-to-Eight-Line Decoder

Inputs			Outputs							
<i>x</i>	<i>y</i>	<i>z</i>	<i>D</i> ₀	<i>D</i> ₁	<i>D</i> ₂	<i>D</i> ₃	<i>D</i> ₄	<i>D</i> ₅	<i>D</i> ₆	<i>D</i> ₇
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Two-to-four-line Decoders with Enable

13



(a) Logic diagram

<i>E</i>	<i>A</i>	<i>B</i>	<i>D</i> ₀	<i>D</i> ₁	<i>D</i> ₂	<i>D</i> ₃
1	<i>X</i>	<i>X</i>	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

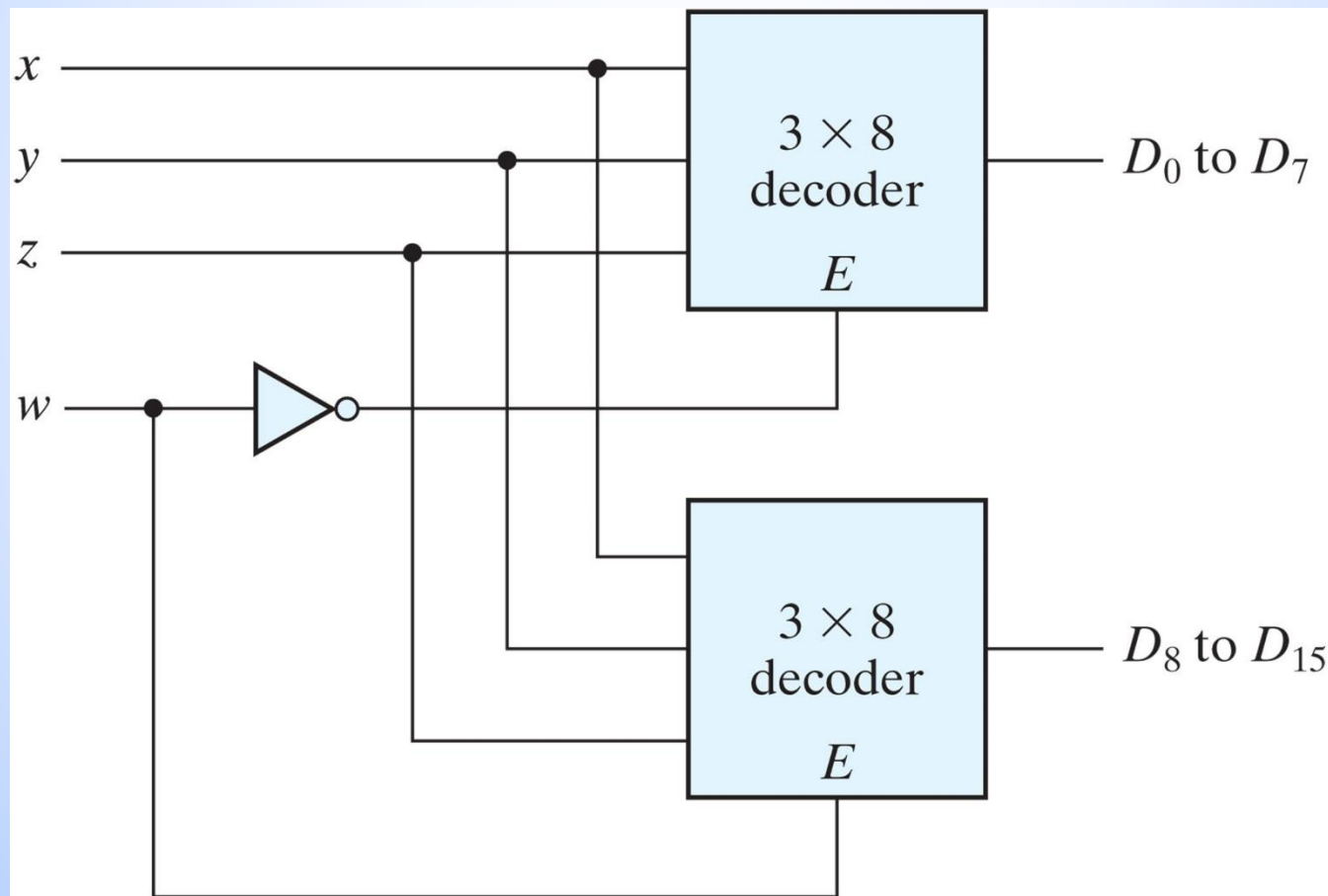
(b) Truth table

Copyright ©2013 Pearson Education, publishing as Prentice Hall

- A decoder **with enable input** can function as a **demultiplexer**
- **Demultiplexer**: A circuit that receives information from a **single line** and directs it to one of **2ⁿ** possible output lines.

4 × 16 decoder constructed with two 3 × 8 decoders

- Decoders with enable inputs can be connected together to form a larger decoder circuit.



Combinational Logic Implementation

15

- Since any Boolean function can be expressed in sum-of-minterms form, a decoder that generates the minterms of the function, together with an external OR gate that forms their logical sum, provides a hardware implementation of the function.

Table 4.4
Full Adder

<i>x</i>	<i>y</i>	<i>z</i>	<i>C</i>	<i>S</i>
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

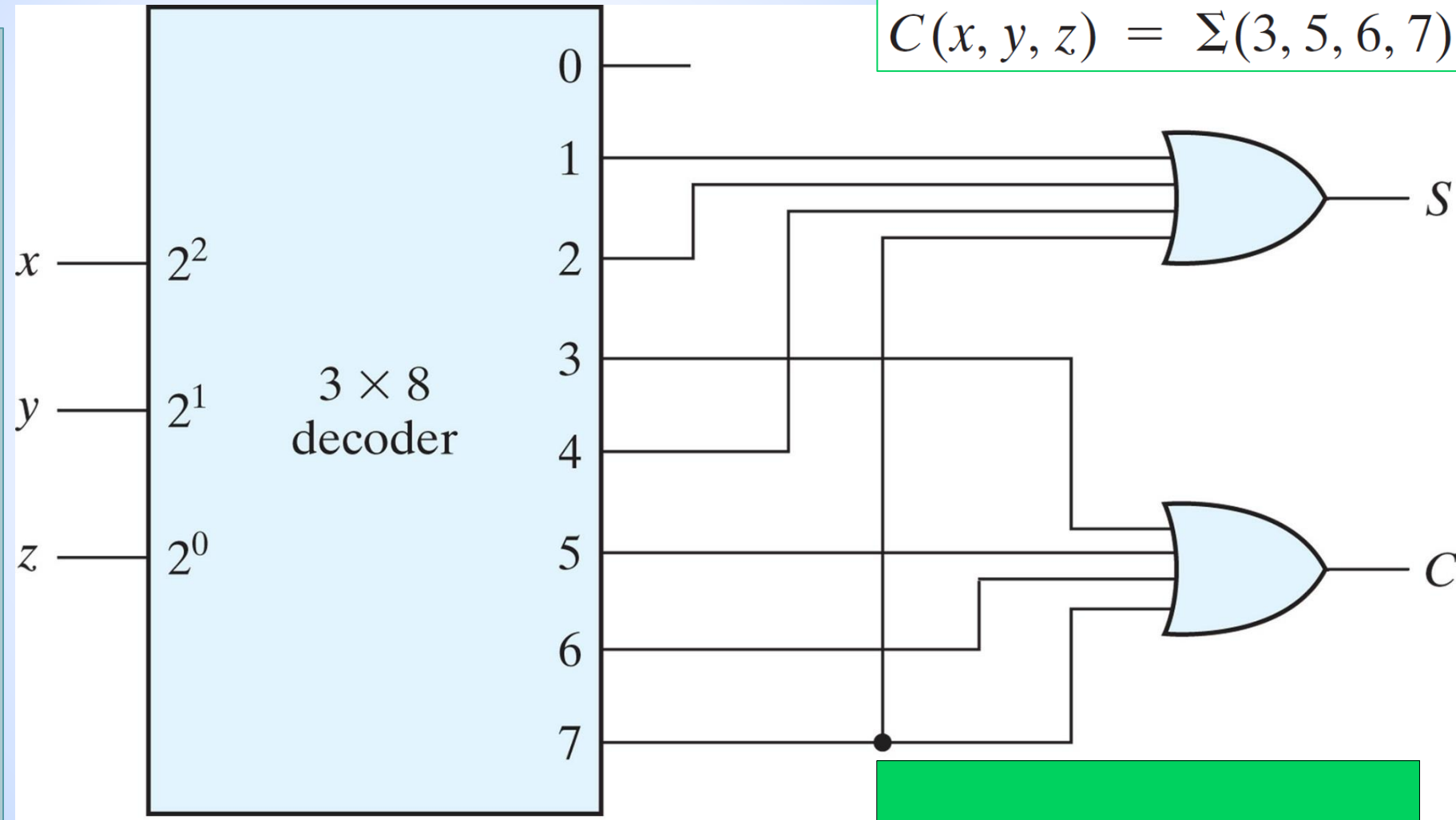
Implementation of a full adder with a decoder

16

- The OR gate for output S forms the logical sum of minterms 1, 2, 4, and 7.
- The OR gate for output C forms the logical sum of minterms 3, 5, 6, and 7.

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$



Implementation of a full adder with a decoder

- ▶ A function with a long list of minterms requires an OR gate with a large number of inputs.
- ▶ A function having a list of k minterms can be expressed in its complemented form F with $2^n - k$ minterms. If the number of minterms in the function is greater than $2^n/2$ then F can be expressed with fewer minterms.
- ▶ In such a case, it is advantageous to use a **NOR gate** to sum the minterms of F .

4.10 Encoders

18

- An encoder is a digital circuit that performs the **inverse operation of a decoder**
- An encoder has 2^n (or fewer) input lines and n output lines
- The output lines, as an aggregate, generate the binary code corresponding to the input value

Octal-to binary Encoder example

19

- The encoder can be implemented with three OR gates

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Table 4.7
Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Octal-to binary Encoder example

20

- The problem:
 - It has the limitation that only one input can be active at any given time
 - If two inputs are active simultaneously, the output produces an undefined combination
- For example, if D_3 and D_6 are 1 simultaneously, the output of the encoder will be 111 because all three outputs are equal to 1
- **The output 111 does not represent either binary 3 or binary 6**
- To resolve this ambiguity, encoder circuits must establish an **input priority** to ensure that **only one input is encoded**
- If both D_3 and D_6 are 1 at the same time, the output will be **110** because D_6 has higher priority than D_3
- **Another ambiguity** in the octal-to-binary encoder is that an output with all 0's is generated when all the inputs are 0; but this output is the same as when D_0 is equal to 1.

Priority Encoder

21

- If two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.
- **Valid** bit indicator is set to 1 when one or more inputs are equal to 1. If all inputs are 0, there is no valid input and V is equal to 0.
- Input D_3 has the highest priority, so, regardless of the values of the other inputs, when this input is 1, the output for xy is 11 (binary 3)

Table 4.8
Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

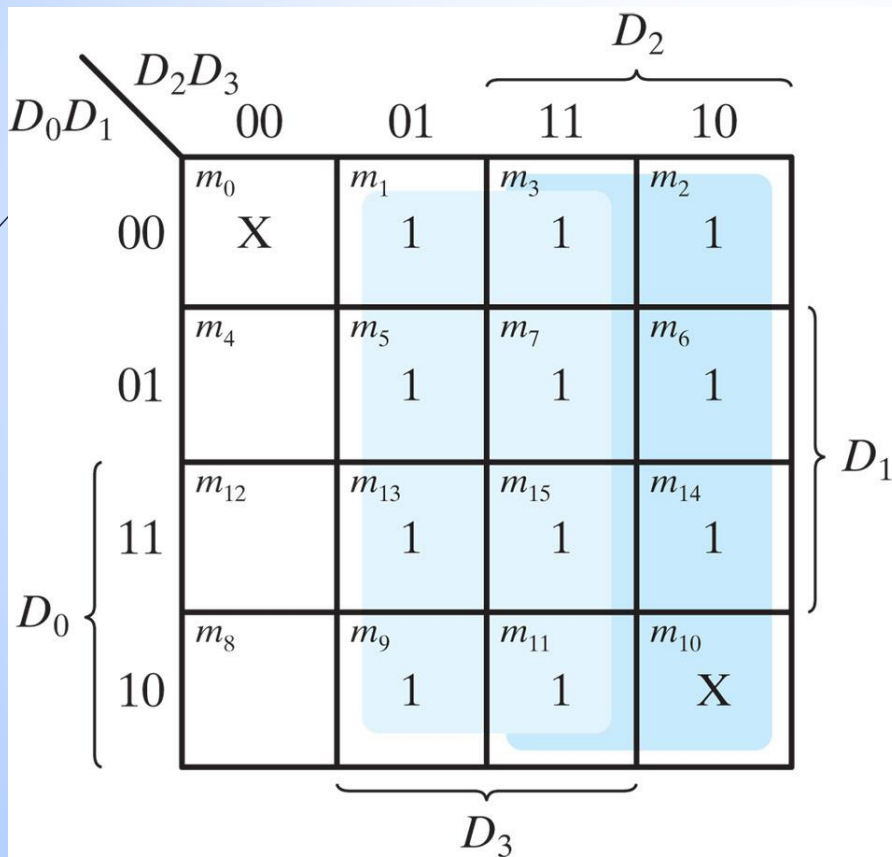
Maps for a Priority Encoder

22

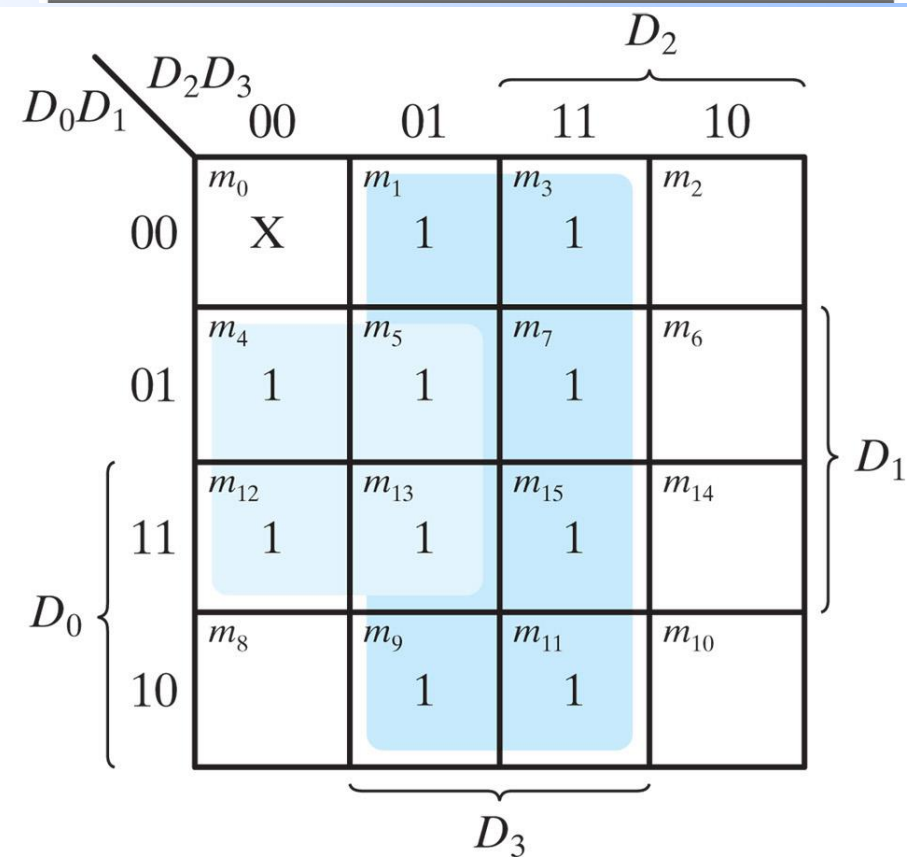
Table 4.8

Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1



$x =$



$y =$



Four-bit Priority Encoder

23

$$x = D_2 + D_3$$

$$y = D_3 + D_1 D_2'$$

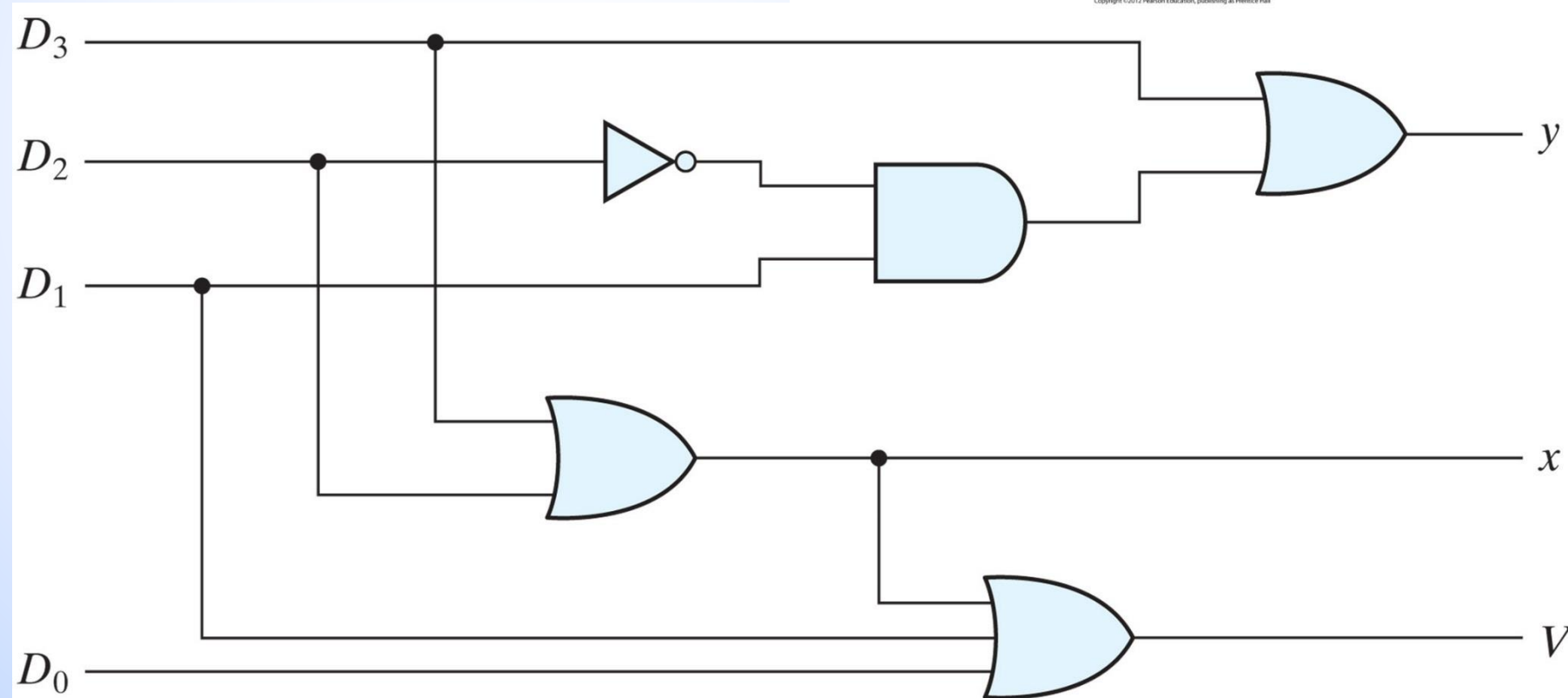
$$V = D_0 + D_1 + D_2 + D_3$$

Table 4.8

Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Copyright ©2012 Pearson Education, publishing as Prentice Hall



Copyright ©2013 Pearson Education, publishing as Prentice Hall

4.11 Multiplexers

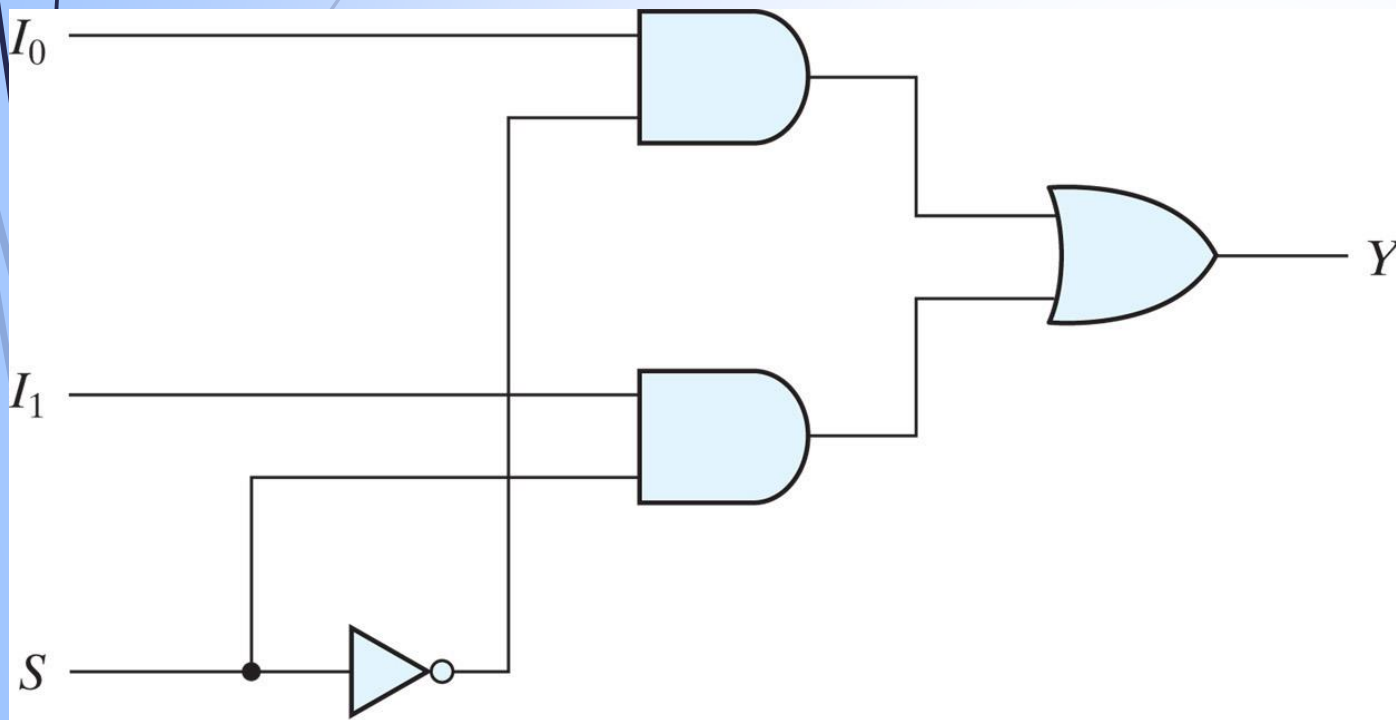
24

- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- There are 2^n input lines and n selection lines whose bit combinations determine which input is selected
- **The multiplexer acts like an electronic switch that selects one of the sources**
- The multiplexer is often labeled “**MUX**” in block diagrams
- A multiplexer is also called a **data selector**

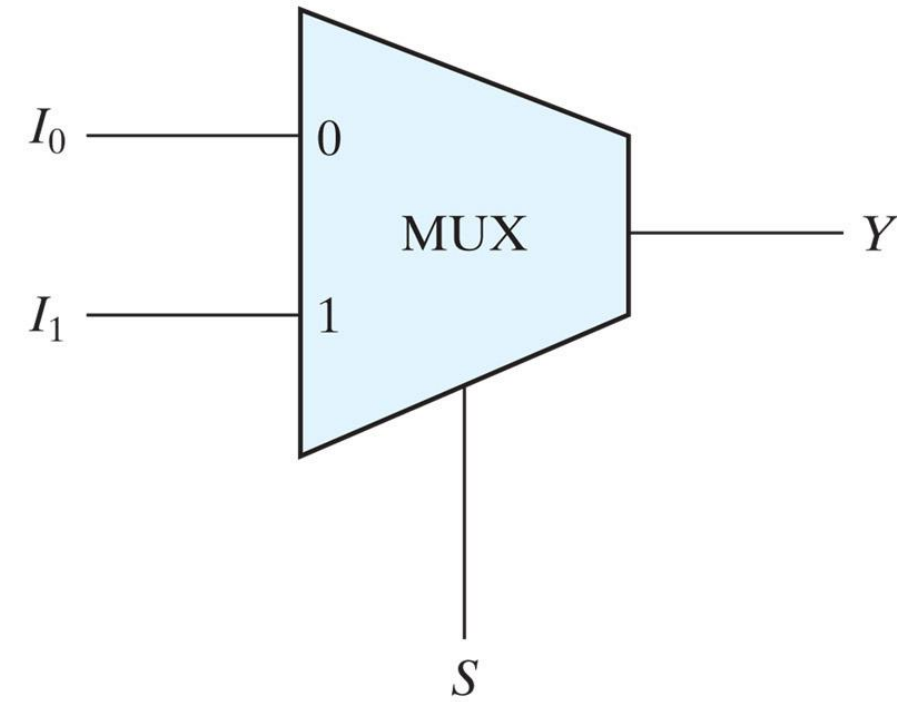
Two-to-one line multiplexer

25

- ▶ A two-to-one-line multiplexer connects one of two 1-bit sources to a common destination



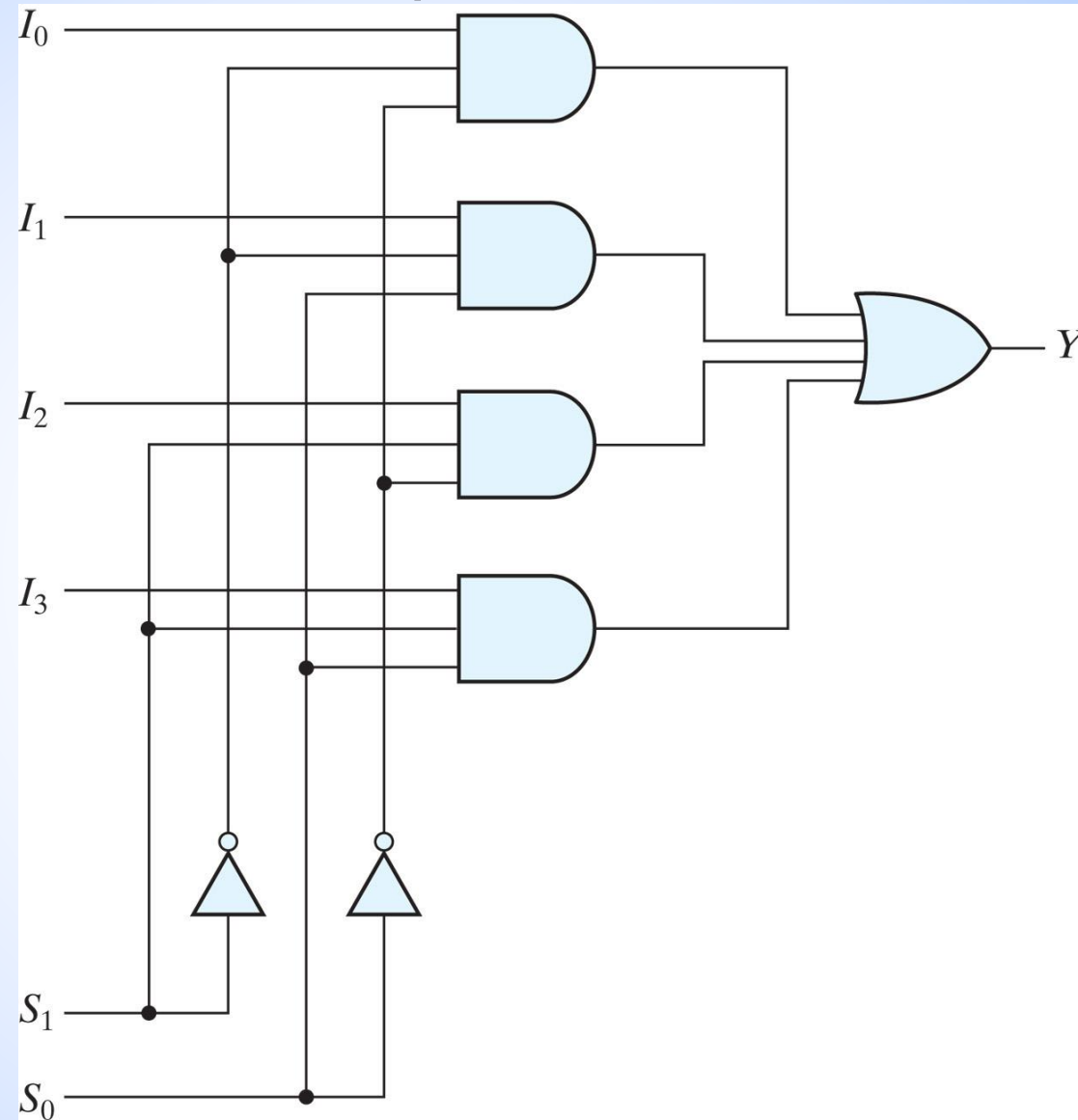
(a) Logic diagram



(b) Block diagram

Four-to-one line multiplexer

26



(a) Logic diagram

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

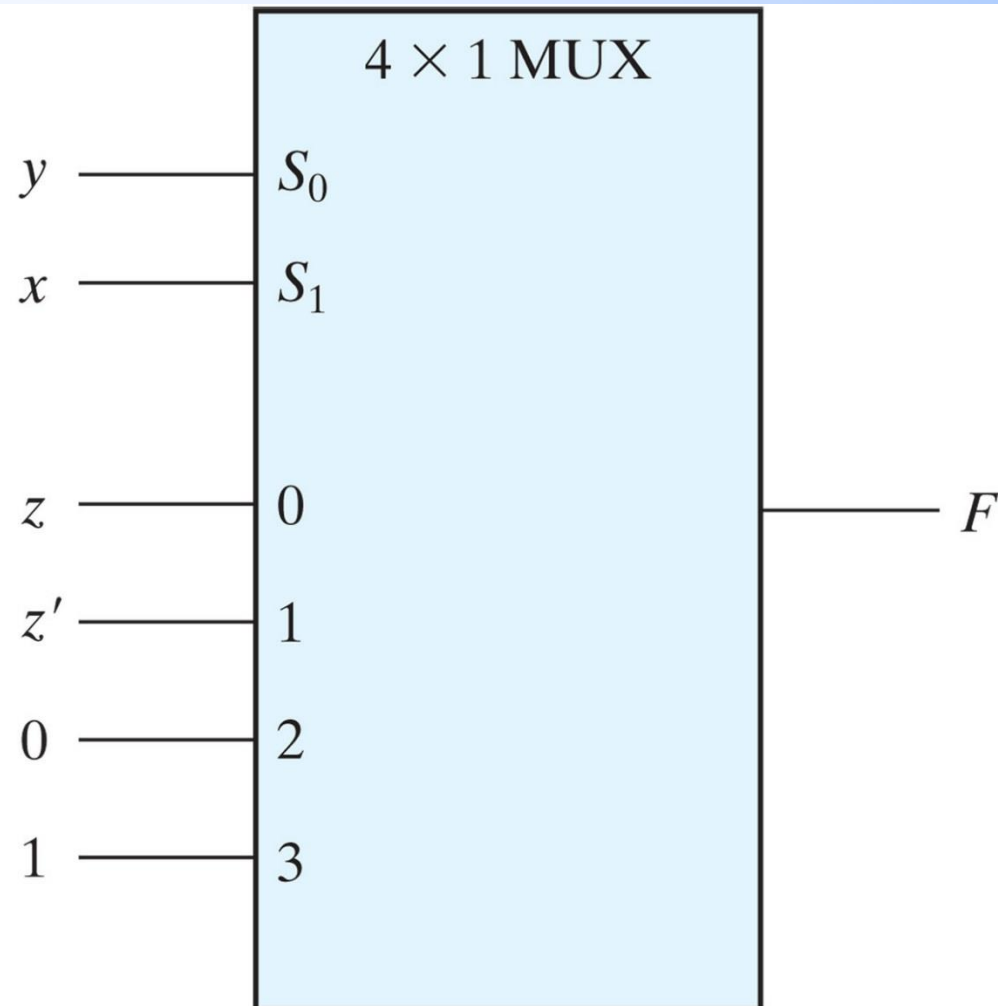
(b) Function table

Implementing a Boolean function with a multiplexer

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

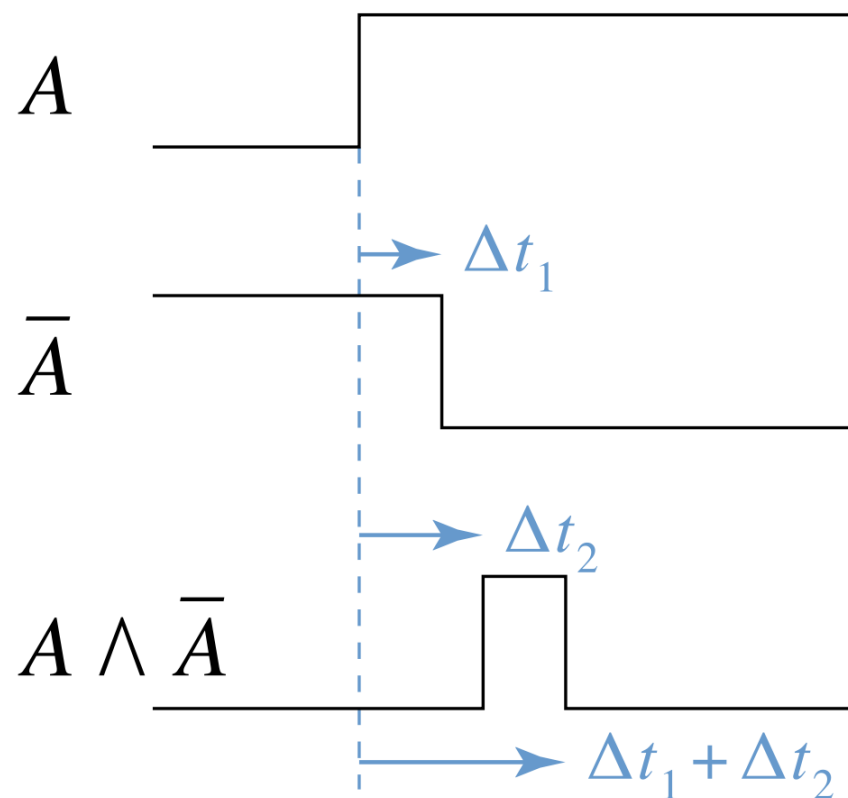
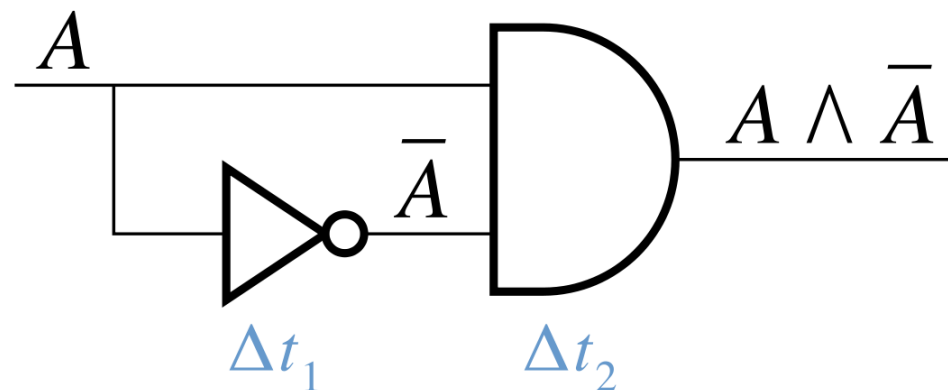


(b) Multiplexer implementation

Hazards

When the input to a combinational logic circuit changes, unwanted switching transients may appear on the output.

These transients occur when different paths from input to output have different propagation delays.



Hazards

- When analyzing combinational logic circuits for hazards we will consider the case where only one input changes at a time.
- Under this condition, a static 1-hazard occurs when the input change causes one product term (in a SOP expression) to transition from 1 to 0 and another product term to transition from 0 to 1.
- Both product terms can be transiently 0, resulting in the static 1-hazard.

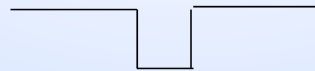
Timing hazard

30

- There is a delay for the logic gates
- When they dd up together they create timing hazards.
 - t_{pLH} = low-to-high, t_{pHL} = high-to-low
- **Three types exist**
 - **Static-0:** Wxpected 0 but received 1 for a short period of time. It occurs in POS circuits
 - **Static 1:** Expected 1 received 0 for a short period of time. It occurs on POS circuits.
 - **Dynamic:** Multiple changes in the output before stable.



Statik 0



Statik 1



Dinamik

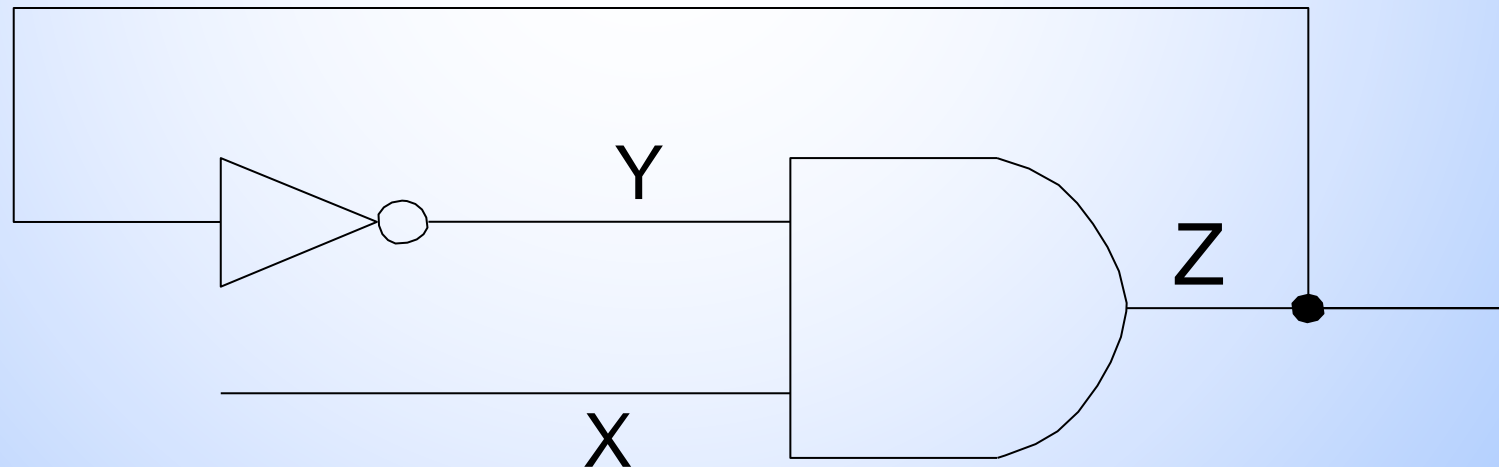
Sample question

31

(inverter) **5ns**, AND gate **10ns**.

Initially $X=0$. X became 1 for 80ns then become 0

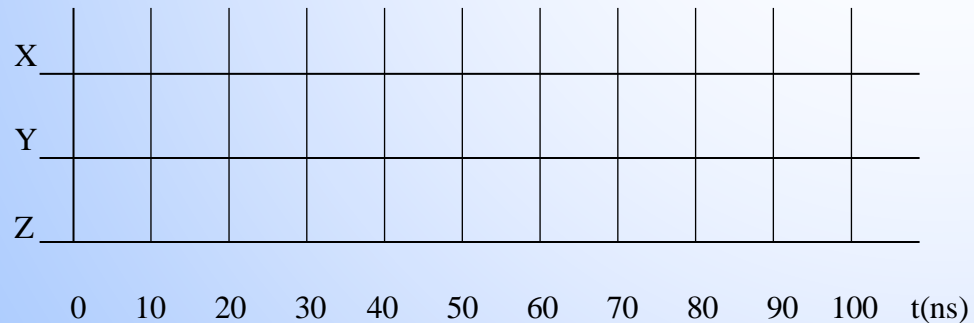
Draw a timing diagram showing changes in X , Y ve Z



Solution

32

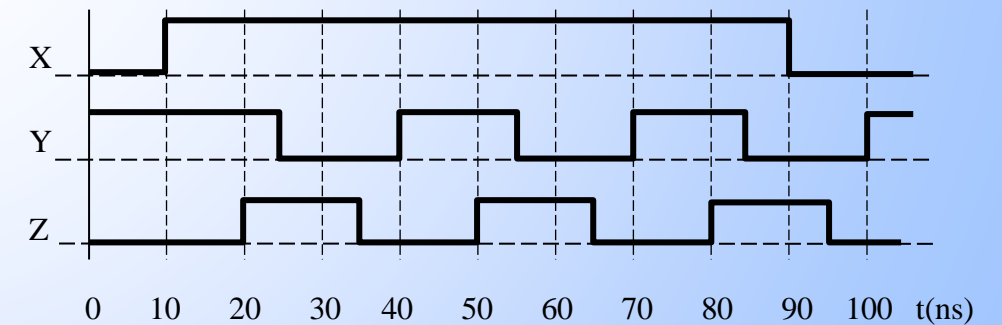
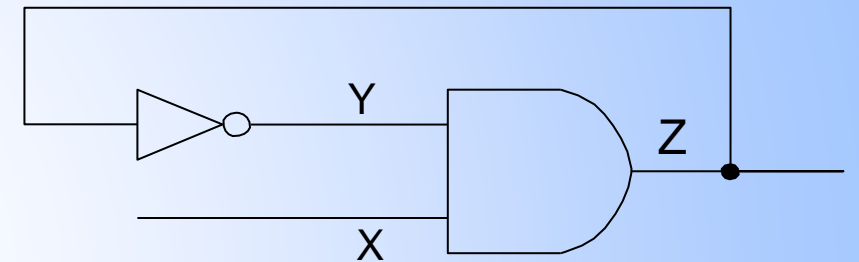
- ▶ Draw an empty timing diagram
- ▶ Show the formula for the circuit
 - ▶ $Z = XY = XZ'$
- ▶ Show initial values of X and Y
- ▶ Draw the rest considering the gate delays



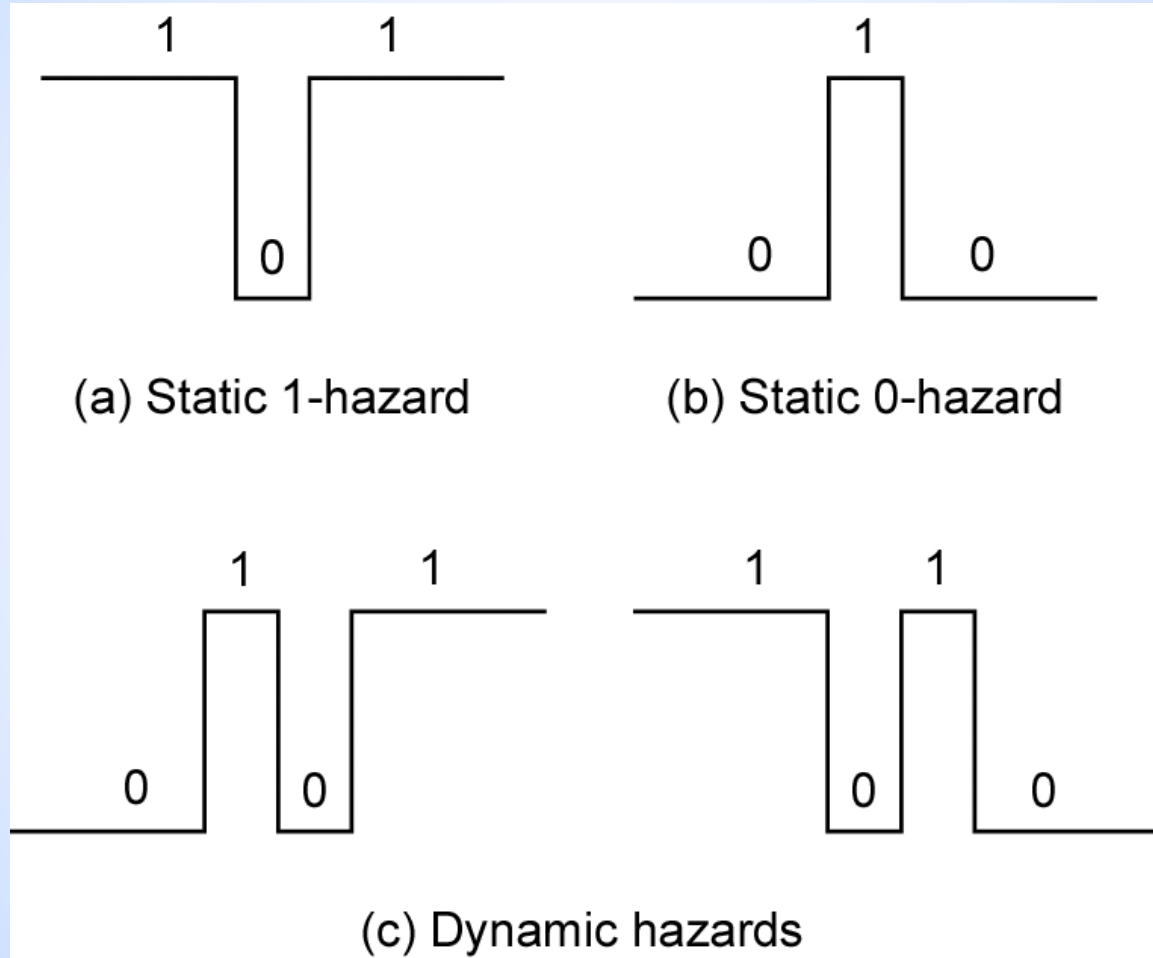
(inverter) **5ns**, AND gate **10ns** delay.

Initially **X=0 Y=1**

What if for **80ns X=1** and then becomes 0



Hazards



Hazards

34

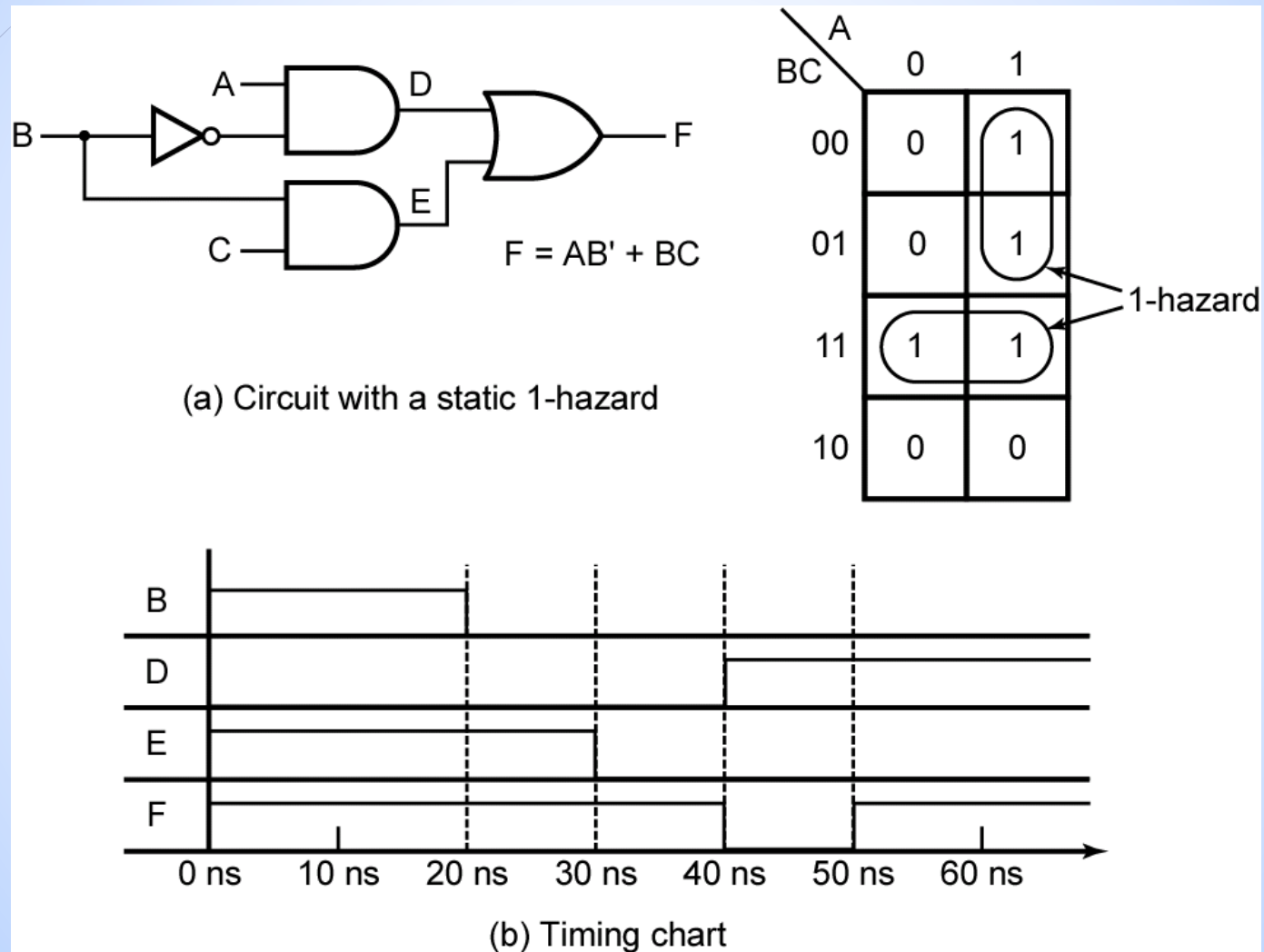
- Under the same condition, a static 0-hazard occurs when the input change causes one sum term (in a POS expression) to transition from 0 to 1 and another sum term to transition from 1 to 0.
- Both sum terms can be transiently 1, resulting in the static 0-hazard.

Detecting Static 1-Hazards

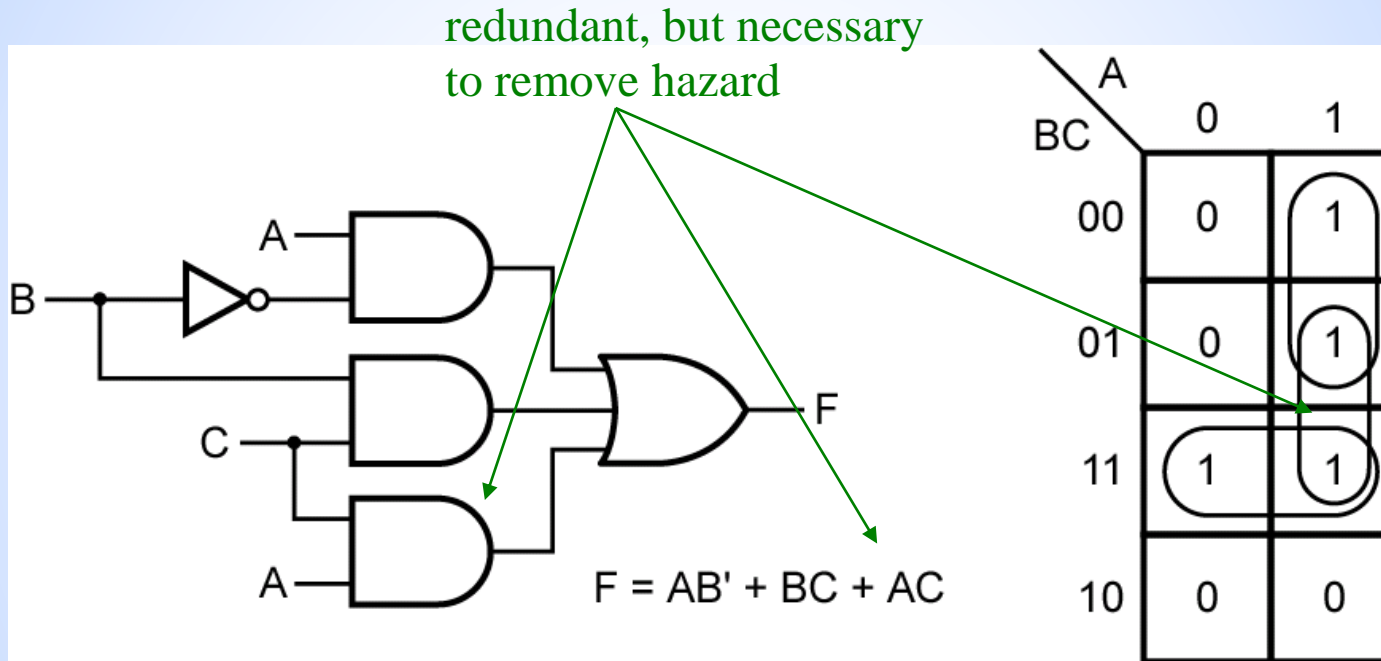
We can detect hazards in a two-level AND-OR circuit using the following procedure:

1. Write down the sum-of-products expression for the circuit.
2. Plot each term on the map and loop it.
3. If any two adjacent 1's are not covered by the same loop, a 1-hazard exists for the transition between the two 1's. For an n -variable map, this transition occurs when one variable changes and the other $n - 1$ variables are held constant.

Detecting Static 1-Hazards



Removing Static 1-Hazards

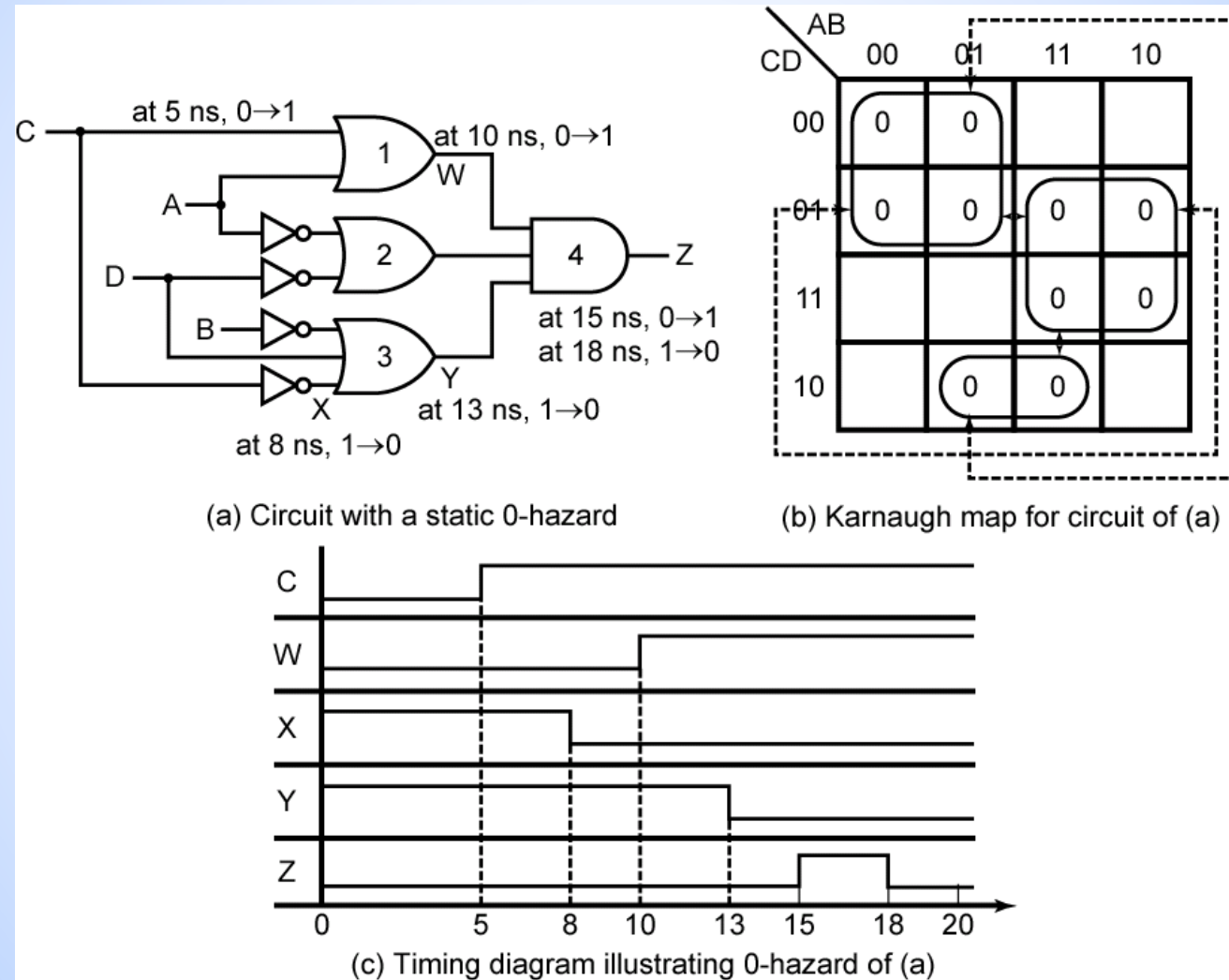


Detecting Static 0-Hazards

We can detect hazards in a two-level OR-AND circuit using the following procedure:

1. Write down the product-of-sums expression for the circuit.
2. Plot each sum term on the map and loop the zeros.
3. If any two adjacent 0's are not covered by the same loop, a 0-hazard exists for the transition between the two 0's. For an n -variable map, this transition occurs when one variable changes and the other $n - 1$ variables are held constant.

Detecting Static 0-Hazards



Removing Static 0-Hazards

40

